```
//@version=5
indicator(title="Insane Oscillator", shorttitle="Insane Oscillator v2.1", format=format.price, precisi

bShowTrend = input.bool(true, "Show trend color", group="Basic Settings")
bShowTramp = input.bool(true, "Show trampoline", group="Basic Settings")
bShowSqueeze = input.bool(true, "Show squeeze dot", group="Basic Settings")
bShowBB = input.bool(true, "Show bollinger bands wicking", group="Basic Settings")
bShowWAE = input.bool(true, "Show waddah explosion", group="Basic Settings")

bShowRSI = input.bool(true, "Show RSI line", group="Basic Settings")
bShowMFI = input.bool(false, "Show MFI line", group="Basic Settings")
bShowVector = input.bool(false, "Show vector candles", group="Basic Settings")

bShowDivies = input.bool(false, "Show divergence background bars", group="Basic Settings")
bShowBuySellies = input.bool(false, "Show buy/sell background bars", group="Basic Settings")
bgTrans = input.int(85, minval=1, title="Background Transparency (100=invisible, 0=opaque)", group="Ba

Theme = input.string(title="Color Theme", options=["Standard", "Pinky and the Brain", "Color Blind", '

thickWaddah = input.int(2, minval=1, title="Waddah Explosion Thickness", group="Basic Settings")
threshold = input.int(120, title="Threshold to trigger (lower is more sensitive)", group="Basic Settir
//bShowHMA = input.bool(false, "Show Hull Moving Average", group="Basic Settings")

rsiLengthInput = input.int(14, minval=1, title="RSI Length", group="RSI Settings")
rsiSourceInput = input.source(close, "Source", group="RSI Settings")
rsiOS = input.int(30, minval=1, title="RSI Oversold", group="RSI Settings")
rsiOB = input.int(70, minval=1, title="RSI Overbought", group="RSI Settings")

rsiLowerW = input.int(30, "Lower Limit of RSI", minval = 1, maxval = 500, group="Wicking Settings")
rsiUpperW = input.int(70, "Upper Limit of RSI", minval = 1, maxval = 500, group="Wicking Settings")

mfiLength = input.int(title="MFI Length", defval=14, minval=1, maxval=2000, group="Money Flow Index")
mfi = ta.mfi(hlc3, mfiLength)

// c1 = Theme == "Standard" ? color1 : Theme == "Classic" ? color3 : Theme == "Lemon" ? color5 : Theme
// c2 = Theme == "Standard" ? color2 : Theme == "Classic" ? color4 : Theme == "Lemon" ? color6 : Theme

colorMildGreen = Theme == "Standard" ? color.new(#66ff00, 40) : Theme == "Pinky and the Brain" ? color
colorBigGreen = Theme == "Standard" ? color.new(#66ff00, 0) : Theme == "Pinky and the Brain" ? color.r

colorMildRed = Theme == "Standard" ? color.new(#ff0000, 40) : Theme == "Pinky and the Brain" ? color.r
colorBigRed = Theme == "Standard" ? color.new(#ff0000, 0) : Theme == "Pinky and the Brain" ? color.new

colorInvisible = color.new(color.black, 100)

upperColor = color.new(#787B86, 0)
lowerColor = color.new(#787B86, 0)
showMeUp = false

// CCI DOUBLE CROSS
ma1 = ta.sma(hlc3, 14)
cci1 = (hlc3 - ma1) / (0.015 * ta.dev(hlc3, 14))
```

```
ma2 = ta.sma(hlc3, 100)
cci2 = (hlc3 - ma1) / (0.015 * ta.dev(hlc3, 100))
cci1WasGreen = cci1[1] > cci2[1] or cci1[2] > cci2[2] or cci1[3] > cci2[3] or cci1[4] > cci2[4] or cci
if (cci1 > cci2 and cci1 >= 100)
    lowerColor := colorBigRed
    showMeUp := true
if (cci1 < cci2 and cci1 < -100 and cci1WasGreen)
    upperColor := colorBigRed

up = ta.rma(math.max(ta.change(rsiSourceInput), 0), rsiLengthInput)
down = ta.rma(-math.min(ta.change(rsiSourceInput), 0), rsiLengthInput)
rsi = down == 0 ? 100 : up == 0 ? 0 : 100 - (100 / (1 + up / down))

mfiColor = #828282
if (mfi > 70)
    mfiColor := color.rgb(0, 148, 76)
if (mfi < 30)
    mfiColor := color.rgb(147, 0, 0)

plot(bShowMFI ? mfi : na, "MFI", color=mfiColor)

lengthHMA = input.int(9, minval=1, title="HMA Length", group="HMA Settings")
srcHMA = input(close, title="HMA Source", group="HMA Settings")
hullma = ta.wma(2*ta.wma(srcHMA, lengthHMA/2)-ta.wma(srcHMA, lengthHMA), math.floor(math.sqrt(lengthHM
hmaColor = hullma > hullma[1] ? colorBigGreen : colorBigRed
//plot(bShowHMA ? hullma : na, color=hmaColor, linewidth=1)


// Divergence
lookbackRight = 5
lookbackLeft = 5
rangeUpper = 60
rangeLower = 5
bearColor = colorMildRed
bullColor = colorMildGreen
textColor = color.white
noneColor = color.new(color.white, 100)

plFound = na(ta.pivotlow(rsi, lookbackLeft, lookbackRight)) ? false : true
phFound = na(ta.pivothigh(rsi, lookbackLeft, lookbackRight)) ? false : true
_inRange(cond) =>
    bars = ta.barssince(cond == true)
    rangeLower <= bars and bars <= rangeUpper

rsiHL = rsi[lookbackRight] > ta.valuewhen(plFound, rsi[lookbackRight], 1) and _inRange(plFound[1])
priceLL = low[lookbackRight] < ta.valuewhen(plFound, low[lookbackRight], 1)
bullCondAlert = priceLL and rsiHL and plFound

rsiLH = rsi[lookbackRight] < ta.valuewhen(phFound, rsi[lookbackRight], 1) and _inRange(phFound[1])
priceHH = high[lookbackRight] > ta.valuewhen(phFound, high[lookbackRight], 1)
bearCondAlert = priceHH and rsiLH and phFound


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   TRAMPOLINE  =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

```
// Idea from "Serious Backtester" - https://www.youtube.com/watch?v=2hX7qTamOAQ
// Defaults are optimized for 30 min candles

// CONFIG
iBBThreshold = input.float(0.0015, minval=0.0, title="Bollinger Lower Threshold", tooltip="0.003 for c
RSIThreshold = input.int(25, minval=1, title="RSI Lower Threshold", tooltip="Normally 25", group="TRAM
RSIDown = input.int(72, minval=1, title="RSI Upper Threshold", tooltip="Normally 75", group="TRAMPOLIN

lengthBB = input.int(20, minval=1, group="TRAMPOLINE Bollinger Bands")
srcBB = input(close, title="Source", group="TRAMPOLINE Bollinger Bands")
multBB = input.float(2.0, minval=0.001, maxval=50, title="StdDev", group="TRAMPOLINE Bollinger Bands")
offsetBB = input.int(0, "Offset", minval = -500, maxval = 500, group="TRAMPOLINE Bollinger Bands")

isRed = close < open
isGreen = close > open

// STANDARD BOLLINGER BANDS
basisBB = ta.sma(srcBB, lengthBB)
devBB = multBB * ta.stdev(srcBB, lengthBB)
upperBB = basisBB + devBB
lowerBB = basisBB - devBB
downBB = low < lowerBB or high < lowerBB
upBB = low > upperBB or high > upperBB
bbw = (upperBB - lowerBB) / basisBB

back1 = isRed[1] and rsi[1] <= RSIThreshold and close[1] < lowerBB[1] and bbw[1] > iBBThreshold
back2 = isRed[2] and rsi[2] <= RSIThreshold and close[2] < lowerBB[2] and bbw[2] > iBBThreshold
back3 = isRed[3] and rsi[3] <= RSIThreshold and close[3] < lowerBB[3] and bbw[3] > iBBThreshold
back4 = isRed[4] and rsi[4] <= RSIThreshold and close[4] < lowerBB[4] and bbw[4] > iBBThreshold
back5 = isRed[5] and rsi[5] <= RSIThreshold and close[5] < lowerBB[5] and bbw[5] > iBBThreshold

for1 = isGreen[1] and rsi[1] >= RSIDown and close[1] > upperBB[1] and bbw[1] > iBBThreshold
for2 = isGreen[2] and rsi[2] >= RSIDown and close[2] > upperBB[2] and bbw[2] > iBBThreshold
for3 = isGreen[3] and rsi[3] >= RSIDown and close[3] > upperBB[3] and bbw[3] > iBBThreshold
for4 = isGreen[4] and rsi[4] >= RSIDown and close[4] > upperBB[4] and bbw[4] > iBBThreshold
for5 = isGreen[5] and rsi[5] >= RSIDown and close[5] > upperBB[5] and bbw[5] > iBBThreshold

weGoUp = isGreen and (back1 or back2 or back3 or back4 or back5) and (high > high[1]) and barstate.isc
upThrust = weGoUp and not weGoUp[1] and not weGoUp[2] and not weGoUp[3] and not weGoUp[4]
weGoDown = isRed and (for1 or for2 or for3 or for4 or for5) and (low < low[1]) and barstate.isconfirme
downThrust = weGoDown and not weGoDown[1] and not weGoDown[2] and not weGoDown[3] and not weGoDown[4]


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-   Squeeze Relaxer version 2.1  =-=-=-=-=-=-=-=-=-=-=-=-=-

// Average Directional Index
sqTolerance = input.int(2, title="Squeeze Tolerance (lower = more sensitive)", group="Relaxing Setting
adxSqueeze = input.int(21, title="ADX Threshold for TTM Squeeze", group="Relaxing Settings", tooltip="

adxlen = input(14, title="ADX Smoothing", group="ADX")
dilen = input(14, title="DI Length", group="ADX")
dirmov(len) =>
    up5 = ta.change(high)
    down5 = -ta.change(low)
    plusDM = na(up5) ? na : (up5 > down5 and up5 > 0 ? up5 : 0)
```

```
    minusDM = na(down5) ? na : (down5 > up5 and down5 > 0 ? down5 : 0)
    truerange = ta.rma(ta.tr, len)
    plus = fixnan(100 * ta.rma(plusDM, len) / truerange)
    minus = fixnan(100 * ta.rma(minusDM, len) / truerange)
    [plus, minus]
adx(dilen, adxlen) =>
    [plus, minus] = dirmov(dilen)
    sum = plus + minus
    adx = 100 * ta.rma(math.abs(plus - minus) / (sum == 0 ? 1 : sum), adxlen)
adxValue = adx(dilen, adxlen)
sigabove19 = adxValue > adxSqueeze

var cGreen = 0
var cRed = 0
var pos = false
var neg = false

sqlength = 20
multQ = 2.0
lengthKC = 20
multKC = 1.5

useTrueRange = true
source = close
basis = ta.sma(source, sqlength)
dev1 = multKC * ta.stdev(source, sqlength)
upperBBsq = basis + dev1
lowerBBsq = basis - dev1
ma = ta.sma(source, lengthKC)
rangeQ = high - low
rangema = ta.sma(rangeQ, lengthKC)
upperKC = ma + rangema * multKC
lowerKC = ma - rangema * multKC
sqzOn  = (lowerBBsq > lowerKC) and (upperBBsq < upperKC)
sqzOff = (lowerBBsq < lowerKC) and (upperBBsq > upperKC)
noSqz  = (sqzOn == false) and (sqzOff == false)

avg1 = math.avg(ta.highest(high, lengthKC), ta.lowest(low, lengthKC))
avg2 = math.avg(avg1, ta.sma(close, lengthKC))
val = ta.linreg(close - avg2, lengthKC, 0)

pos := false
neg := false

// if squeeze is bright RED, increment by one
if (val < nz(val[1]) and val < 5 and not sqzOn)
    cRed := cRed + 1

// if squeeze is bright GREEN, increment by one
if (val > nz(val[1]) and val > 5 and not sqzOn)
    cGreen := cGreen + 1

// if bright RED squeeze is now dim, momentum has changed.  Is ADX also above 19? - add a marker to ch
if (val > nz(val[1]) and cRed > sqTolerance and val < 5 and not pos[1] and sigabove19 == true)
    cRed := 0
```

```
    pos := true

// if bright GREEN squeeze is now dim, momentum has changed.  Is ADX also above 19? - add a marker to
if (val < nz(val[1]) and cGreen > sqTolerance and val > 5 and not neg[1] and sigabove19 == true)
    cGreen := 0
    neg := true

buySignal1 = pos and barstate.isconfirmed
sellSignal1 = neg and barstate.isconfirmed


// JOHN WICK BOLLINGER BANDS
wlengthBB = input.int(20, minval=1, group="Wicking Bollinger Bands")
wsrcBB = input(close, title="Source", group="Wicking Bollinger Bands")
wmultBB = input.float(2.5, minval=0.001, maxval=50, title="StdDev", group="Wicking Bollinger Bands")
woffsetBB = input.int(0, "Offset", minval = -500, maxval = 500, group="Wicking Bollinger Bands")
wbasisBB = ta.sma(wsrcBB, wlengthBB)
wdevBB = wmultBB * ta.stdev(wsrcBB, wlengthBB)
wupperBB = wbasisBB + wdevBB
wlowerBB = wbasisBB - wdevBB


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   VECTOR CANDLES  =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

import TradersReality/Traders_Reality_Lib/2 as trLib

color redVectorColor = colorBigRed
color greenVectorColor = colorBigGreen
color violetVectorColor = color.fuchsia
color blueVectorColor = color.rgb(83, 144, 249)
color regularCandleUpColor = color.new(#999999, 99)
color regularCandleDownColor = color.new(#4d4d4d, 99)

bool overrideSym = false
string pvsraSym = 'INDEX:BTCUSD'
bool colorOverride = true

pvsraVolume(overrideSymbolX, pvsraSymbolX, tickerIdX) =>
    request.security(overrideSymbolX ? pvsraSymbolX : tickerIdX, '', [volume,high,low,close,open], bar
[pvsraVolume, pvsraHigh, pvsraLow, pvsraClose, pvsraOpen]  = pvsraVolume(overrideSym, pvsraSym, syminf
[pvsraColor, alertFlag, averageVolume, volumeSpread, highestVolumeSpread] = trLib.calcPvsra(pvsraVolum


// IMPULSE MACD
calc_smma(src, len) =>
    smma = 0.0
    smma := na(smma[1]) ? ta.sma(src, len) : (smma[1] * (len - 1) + src) / len
    smma

calc_zlema(src, length) =>
    ema1 = ta.ema(src, length)
    ema2 = ta.ema(ema1, length)
    d = ema1 - ema2
    ema1 + d
```

```
srcMCD = hlc3
hiMCD = calc_smma(high, 34)
loMCD = calc_smma(low, 34)
miMCD = calc_zlema(srcMCD, 34)
mdMCD = (miMCD>hiMCD) ? (miMCD-hiMCD) : (miMCD<loMCD) ? (miMCD-loMCD) : 0
sbMCD = ta.sma(mdMCD, 9)
shMCD = mdMCD - sbMCD


// TRIPLE SUPERTREND
atr25 = ta.sma(ta.tr, 10)
atr= atr25
tx=hl2-(1*atr)
tx1 = nz(tx[1],tx)
tx := close[1] > tx1 ? math.max(tx,tx1) : tx
ty=hl2+(1*atr)
ty1 = nz(ty[1], ty)
ty := close[1] < ty1 ? math.min(ty, ty1) : ty
trend5 = 1
trend5 := nz(trend5[1], trend5)
trend5 := trend5 == -1 and close > ty1 ? 1 : trend5 == 1 and close < tx1 ? -1 : trend5
changeCond = trend5 != trend5[1]

atr20 = ta.sma(ta.tr, 11)
atr0 = atr20
tx0 =hl2-(2*atr)
tx10 = nz(tx0[1],tx0)
tx0 := close[1] > tx10 ? math.max(tx0,tx10) : tx0
ty0=hl2+(2*atr)
ty10 = nz(ty0[1], ty0)
ty0 := close[1] < ty10 ? math.min(ty0, ty10) : ty0
trend0 = 1
trend0 := nz(trend0[1], trend0)
trend0 := trend0 == -1 and close > ty10 ? 1 : trend0 == 1 and close < tx10 ? -1 : trend0
changeCond0 = trend0 != trend0[1]

atr29 = ta.sma(ta.tr, 12)
atr9 = atr29
tx9=hl2-(3*atr)
tx19 = nz(tx9[1],tx9)
tx9 := close[1] > tx19 ? math.max(tx9,tx19) : tx9
ty9=hl2+(3*atr)
ty19 = nz(ty9[1], ty9)
ty9 := close[1] < ty19 ? math.min(ty9, ty19) : ty9
trend9 = 1
trend9 := nz(trend9[1], trend9)
trend9 := trend9 == -1 and close > ty19 ? 1 : trend9 == 1 and close < tx19 ? -1 : trend9
changeCond9 = trend9 != trend9[1]

var tripBuy = false
var tripSell = false

tripBuy := (trend9==1 and tx9 and trend5==1 and tx and trend0==1 and tx0) or (trend9==1 and tx9 and tr
tripSell := (trend9!=1 and ty9 and trend5!=1 and ty and trend0!=1 and ty0) or (trend9!=1 and ty9 and t
```

```
// Halftrend
amplitude = input(title="Amplitude", defval=2, group="Halftrend")
channelDeviation = input(title="Channel Deviation", defval=2, group="Halftrend")

var int trend = 0
var int nextTrend = 0
var float maxLowPrice = nz(low[1], low)
var float minHighPrice = nz(high[1], high)

var float up1 = 0.0
var float down1 = 0.0
float atrHigh = 0.0
float atrLow = 0.0
float arrowUp = na
float arrowDown = na

atr2 = ta.atr(100) / 2
dev = channelDeviation * atr2

highPrice = high[math.abs(ta.highestbars(amplitude))]
lowPrice = low[math.abs(ta.lowestbars(amplitude))]
highma = ta.sma(high, amplitude)
lowma = ta.sma(low, amplitude)

if nextTrend == 1
    maxLowPrice := math.max(lowPrice, maxLowPrice)

    if highma < maxLowPrice and close < nz(low[1], low)
        trend := 1
        nextTrend := 0
        minHighPrice := highPrice
else
    minHighPrice := math.min(highPrice, minHighPrice)

    if lowma > minHighPrice and close > nz(high[1], high)
        trend := 0
        nextTrend := 1
        maxLowPrice := lowPrice

if trend == 0
    if not na(trend[1]) and trend[1] != 0
        up1 := na(down1[1]) ? down1 : down1[1]
        arrowUp := up1 - atr2
    else
        up1 := na(up1[1]) ? maxLowPrice : math.max(maxLowPrice, up1[1])
    atrHigh := up1 + dev
    atrLow := up1 - dev
else
    if not na(trend[1]) and trend[1] != 1
        down1 := na(up1[1]) ? up1 : up1[1]
        arrowDown := down1 + atr2
    else
        down1 := na(down1[1]) ? minHighPrice : math.min(minHighPrice, down1[1])
    atrHigh := down1 + dev
    atrLow := down1 - dev
```

```
HalfTrue = trend == 0
htColor = trend == 0 ? colorBigGreen : colorBigRed


// BULL RUSH
bullUp = (ta.ema(close, 9) > ta.ema(close, 21) and close > ta.ema(close, 50) and open > ta.ema(close,
bullDown = (ta.ema(close, 9) < ta.ema(close, 21) and close < ta.ema(close, 50) and open < ta.ema(close


// KNOW SURE THING
roclen1 = 10
roclen2 = 15
roclen3 = 20
roclen4 = 30
smalen1 = 10
smalen2 = 10
smalen3 = 10
smalen4 = 15
siglen = 9
smaroc(roclen, smalen) => ta.sma(ta.roc(close, roclen), smalen)
kst = smaroc(roclen1, smalen1) + 2 * smaroc(roclen2, smalen2) + 3 * smaroc(roclen3, smalen3) + 4 * sma
sig = ta.sma(kst, siglen)
KSTTrue = kst >= sig


// ====================================   VODKA SHOT   ============================================


var isLong = false
var isShort = false

// ==================================  NEGLECTED VOL by DGT  ======================================--

nzVolume = nz(volume)
source5   = barstate.isconfirmed ? close : close[1]
vsource  = nzVolume ? barstate.isconfirmed ? ta.obv : ta.obv[1] : na
corr     = ta.correlation(source5, vsource, 14)
volAvgS  = ta.sma(nzVolume, 14)
volAvgL  = ta.sma(nzVolume, 14 * 5)
volDev   = (volAvgL + 1.618034 * ta.stdev(volAvgL, 14 * 5)) / volAvgL * 11 / 100
volRel   = nzVolume / volAvgL
momentum = ta.change(vsource, 14) / 14
momOsc   = ta.linreg(momentum / volAvgS * 1.618034, 5, 0)

vbcbColor = if close < open
    if nzVolume > volAvgS * 1.618034
        #910000
    else if nzVolume >= volAvgS * .618034 and nzVolume <= volAvgS * 1.618034
        color.red
    else
        color.orange
else
    if nzVolume > volAvgS * 1.618034
        #006400
    else if nzVolume >= volAvgS * .618034 and nzVolume <= volAvgS * 1.618034
```

```
        color.green
    else
        #7FFFD4

bColor5 = color.new(color.black, 25)
gColor = color.new(color.gray, 50)

// ==================================  RedK Dual VADER with Energy Bars [VADER-DEB]  ================

f_derma(_data, _len, MAOption) =>
    value =
      MAOption == 'SMA' ? ta.sma(_data, _len) :
      MAOption == 'EMA' ? ta.ema(_data, _len) :
      ta.wma(_data, _len)

rlength  = input.int(12, minval=1)
DER_avg = input.int(5, 'Average', minval=1, inline='DER', group='RedK Dual VADER')
MA_Type5 = input.string('WMA', 'DER MA type', options=['WMA', 'EMA', 'SMA'], inline='DER', group='RedK
rsmooth  = input.int(3, 'Smooth', minval=1,  inline='DER_1', group='RedK Dual VADER')

show_senti = input.bool(true, 'Sentiment',  inline='DER_s', group='RedK Dual VADER')
senti   = input.int(20, 'Length', minval=1, inline='DER_s', group='RedK Dual VADER')

v_calc  = input.string('Relative', 'Calculation', options=['Relative', 'Full', 'None'], group='RedK Du
vlookbk = input.int(20, 'Lookback (for Relative)', minval=1,                            group='RedK Du

v5 = volume

vola    =
  v_calc == 'None' or na(volume) ? 1 :
  v_calc == 'Relative' ?   ta.stoch(v5, v5, v5, vlookbk) / 100 :
  v5

R       = (ta.highest(2) - ta.lowest(2)) / 2                  // R is the 2-bar average bar range -
sr      = ta.change(close) / R                                // calc ratio of change to R
rsr     = math.max(math.min(sr, 1), -1)                       // ensure ratio is restricted to +1/-1
c       = fixnan(rsr * vola)

c_plus  = math.max(c, 0)                                      // calc directional vol-accel energy
c_minus = -math.min(c, 0)

avg_vola    = f_derma(vola, rlength, MA_Type5)
dem         = f_derma(c_plus, rlength, MA_Type5)  / avg_vola         // directional energy ratio
sup         = f_derma(c_minus, rlength, MA_Type5) / avg_vola

adp         = 100 * ta.wma(nz(dem), DER_avg)                     // average DER
asp         = 100 * ta.wma(nz(sup), DER_avg)
anp         = adp - asp                                          // net DER..
anp_s       = ta.wma(anp, rsmooth)

s_adp       = 100 * ta.wma(nz(dem), senti)                         // average DER for sentiment ler
s_asp       = 100 * ta.wma(nz(sup), senti)
V_senti     = ta.wma(s_adp - s_asp, rsmooth)

c_adp   = color.new(#11ff20, 30)
```

```
c_asp   = color.new(#ff1111, 30)
c_fd    = color.new(color.green, 80)
c_fs    = color.new(color.red, 80)
c_zero  = color.new(#ffee00, 70)

c_up5    = color.new(#11ff20, 0)
c_dn5    = color.new(#ff1111, 0)

up5      = anp_s >= 0
s_up    = V_senti >= 0

c_grow_above = #1b5e2080
c_grow_below = #dc4c4a80
c_fall_above = #66bb6a80
c_fall_below = #ef8e9880

sflag_up = math.abs(V_senti) >= math.abs(V_senti[1])

bo = fixnan(asp)
bc = fixnan(adp)
bh = math.max(bo, bc)
bl = math.min(bo, bc)

rising      = ta.change(bc) > 0

c_barup     = #11ff2088
c_bardn     = #ff111188
c_bardj     = #ffffff88

barcolor    = bc > bo and rising ? c_barup : bc < bo and not rising ? c_bardn : c_bardj

// ==================================  RedK Slow_Smooth WMA, RSS_WMA v3 ============================

f_LazyLine(_data, _length) =>
    w1 = 0, w2 = 0, w3 = 0
    L1 = 0.0, L2 = 0.0, L3 = 0.0
    w = _length / 3

    if _length > 2
        w2 := math.round(w)
        w1 := math.round((_length-w2)/2)
        w3 :=   int((_length-w2)/2)

        L1 := ta.wma(_data, w1)
        L2 := ta.wma(L1, w2)
        L3 := ta.wma(L2, w3)
    else
        L3 := _data
    L3

LL = f_LazyLine(close, 21)

lc_up        = color.new(#33ff00, 0)
lc_dn        = color.new(#ff1111, 0)
luptrend     = LL > LL[1]
```

```
SigMulti    = 1.0

SignalOn    = barstate.isconfirmed
SwingDn     = luptrend[1] and not(luptrend) and barstate.isconfirmed
SwingUp     = luptrend and not(luptrend[1]) and barstate.isconfirmed

dl = SigMulti / 100 * LL

upwards = LL > LL[1] and (barcolor == c_barup) and up and (open < close and volRel * .145898 > volDev)
downwards = LL < LL[1]  and (barcolor == c_bardn) and (open > close and volRel * .145898 > volDev)



// WAE - Waddah Attar Explosion v1 by LazyBear

sensitivity = input.int(150, title="Sensitivity", group="WAE")
fastLength=input.int(20, title="FastEMA Length", group="WAE")
slowLength=input.int(40, title="SlowEMA Length", group="WAE")
channelLength=input.int(20, title="BB Channel Length", group="WAE")
multWAE=input.float(2.0, title="BB Stdev Multiplier", group="WAE")

calc_macd(source, fastLength, slowLength) =>
    fastMA = ta.ema(source, fastLength)
    slowMA = ta.ema(source, slowLength)
    fastMA - slowMA

calc_BBUpper(source, length, mult) =>
    basis = ta.sma(source, length)
    dev = mult * ta.stdev(source, length)
    basis + dev

calc_BBLower(source, length, mult) =>
    basis = ta.sma(source, length)
    dev = mult * ta.stdev(source, length)
    basis - dev

t1 = (calc_macd(close, fastLength, slowLength) - calc_macd(close[1], fastLength, slowLength))*sensitiv
e1 = (calc_BBUpper(close, channelLength, multWAE) - calc_BBLower(close, channelLength, multWAE))

trendUpWAE = (t1 >= 0) ? t1 : 0
trendDownWAE = (t1 < 0) ? (-1*t1) : 0

waeColor = trendUpWAE >= threshold ? colorBigGreen : trendDownWAE >= threshold ? colorBigRed : color.r


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   SHARK WAVE TREND  =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

wtChannelLen = input.int(9, title = 'WT Channel Length', group="SharkWaveTrend")
wtAverageLen = input.int(12, title = 'WT Average Length', group="SharkWaveTrend")
wtMASource = input.source(hlc3, title = 'WT MA Source', group="SharkWaveTrend")
wtMALen = input.int(3, title = 'WT MA Length', group="SharkWaveTrend")

c_intense_red = colorBigRed
c_regular_red = colorMildRed
c_intense_green = colorBigGreen
```

```
c_regular_green = colorMildGreen

obLevel = input.int(53, title = 'WT Overbought Level 1', group="SharkWaveTrend")
osLevel = input.int(-53, title = 'WT Oversold Level 1', group="SharkWaveTrend")

wtDivOBLevel = input.int(45, title = 'WT Bearish Divergence min', group="SharkWaveTrend")
wtDivOSLevel = input.int(-65, title = 'WT Bullish Divergence min', group="SharkWaveTrend")

wtDivOBLevel_add = input.int(15, title = 'WT 2nd Bearish Divergence', group="SharkWaveTrend")
wtDivOSLevel_add = input.int(-40, title = 'WT 2nd Bullish Divergence 15 min', group="SharkWaveTrend")

rsiMFIperiod = input.int(60,title = 'MFI Period', group="SharkWaveTrend")
rsiMFIMultiplier = input.int(150, title = 'MFI Area multiplier', group="SharkWaveTrend")
rsiMFIPosY = input.float(2.5, title = 'MFI Area Y Pos', group="SharkWaveTrend")

rsiSRC = input.source(close, title = 'RSI Source', group="SharkWaveTrend")
rsiLen = input.int(14, title = 'RSI Length', group="SharkWaveTrend")
rsiOversold = input.int(30, title = 'RSI Oversold', minval = 0, maxval = 50, group="SharkWaveTrend")
rsiOverbought = input.int(60, title = 'RSI Overbought', minval = 50, maxval = 100, group="SharkWaveTre

rsiDivOBLevel = input.int(60, title = 'RSI Bearish Divergence min', group="SharkWaveTrend")
rsiDivOSLevel = input.int(30, title = 'RSI Bullish Divergence min', group="SharkWaveTrend")

stochSRC = input.source(close, title = 'Stochastic RSI Source', group="SharkWaveTrend")
stochLen = input.int(14, title = 'Stochastic RSI Length', group="SharkWaveTrend")
stochRsiLen = input.int(14, title = 'RSI Length ', group="SharkWaveTrend")
stochKSmooth = input.int(3, title = 'Stochastic RSI K Smooth', group="SharkWaveTrend")
stochDSmooth = input.int(3, title = 'Stochastic RSI D Smooth', group="SharkWaveTrend")

colorRedWT = colorBigRed
colorGreenWT = colorBigGreen

f_top_fractal(src) => src[4] < src[2] and src[3] < src[2] and src[2] > src[1] and src[2] > src[0]
f_bot_fractal(src) => src[4] > src[2] and src[3] > src[2] and src[2] < src[1] and src[2] < src[0]
f_fractalize(src) => f_top_fractal(src) ? 1 : f_bot_fractal(src) ? -1 : 0

f_findDivs(src, topLimit, botLimit, useLimits) =>
    fractalTop = f_fractalize(src) > 0 and (useLimits ? src[2] >= topLimit : true) ? src[2] : na
    fractalBot = f_fractalize(src) < 0 and (useLimits ? src[2] <= botLimit : true) ? src[2] : na
    highPrev = ta.valuewhen(fractalTop, src[2], 0)[2]
    highPrice = ta.valuewhen(fractalTop, high[2], 0)[2]
    lowPrev = ta.valuewhen(fractalBot, src[2], 0)[2]
    lowPrice = ta.valuewhen(fractalBot, low[2], 0)[2]
    bearSignal = fractalTop and high[2] > highPrice and src[2] < highPrev
    bullSignal = fractalBot and low[2] < lowPrice and src[2] > lowPrev
    bearDivHidden = fractalTop and high[2] < highPrice and src[2] > highPrev
    bullDivHidden = fractalBot and low[2] > lowPrice and src[2] < lowPrev
    [fractalTop, fractalBot, lowPrev, bearSignal, bullSignal, bearDivHidden, bullDivHidden]

f_rsimfi(_period, _multiplier, _tf) => request.security(syminfo.tickerid, _tf, ta.sma(((close - open)

f_wavetrend(src, chlen, avg, malen, tf) =>
    tfsrc = request.security(syminfo.tickerid, tf, src)
    esa = ta.ema(tfsrc, chlen)
    de = ta.ema(math.abs(tfsrc - esa), chlen)
```

```
    ci = (tfsrc - esa) / (0.015 * de)
    wt1 = request.security(syminfo.tickerid, tf, ta.ema(ci, avg))
    wt2 = request.security(syminfo.tickerid, tf, ta.sma(wt1, malen))
    wtVwap = wt1 - wt2
    wtOversold = wt2 <= osLevel
    wtOverbought = wt2 >= obLevel
    wtCross = ta.cross(wt1, wt2)
    wtCrossUp = wt2 - wt1 <= 0
    wtCrossDown = wt2 - wt1 >= 0
    wtCrosslast = ta.cross(wt1[2], wt2[2])
    wtCrossUplast = wt2[2] - wt1[2] <= 0
    wtCrossDownlast = wt2[2] - wt1[2] >= 0
    [wt1, wt2, wtOversold, wtOverbought, wtCross, wtCrossUp, wtCrossDown, wtCrosslast, wtCrossUplast,

f_stochrsi(_src, _stochlen, _rsilen, _smoothk, _smoothd, _log, _avg) =>
    src = _log ? math.log(_src) : _src
    rsiWT = ta.rsi(src, _rsilen)
    kk = ta.sma(ta.stoch(rsiWT, rsiWT, rsiWT, _stochlen), _smoothk)
    d1 = ta.sma(kk, _smoothd)
    avg_1 = math.avg(kk, d1)
    k = _avg ? avg_1 : kk
    [k, d1]

rsiWT = ta.rsi(rsiSRC, rsiLen)

[wt1, wt2, wtOversold, wtOverbought, wtCross, wtCrossUp, wtCrossDown, wtCross_last, wtCrossUp_last, wt

[stochK, stochD] = f_stochrsi(stochSRC, stochLen, stochRsiLen, stochKSmooth, stochDSmooth, true, true)

[wtFractalTop, wtFractalBot, wtLow_prev, wtBearDiv, wtBullDiv, wtBearDivHidden, wtBullDivHidden] = f_f

[wtFractalTop_add, wtFractalBot_add, wtLow_prev_add, wtBearDiv_add, wtBullDiv_add, wtBearDivHidden_adc
[wtFractalTop_nl, wtFractalBot_nl, wtLow_prev_nl, wtBearDiv_nl, wtBullDiv_nl, wtBearDivHidden_nl, wtBu

[rsiFractalTop, rsiFractalBot, rsiLow_prev, rsiBearDiv, rsiBullDiv, rsiBearDivHidden, rsiBullDivHidder
[rsiFractalTop_nl, rsiFractalBot_nl, rsiLow_prev_nl, rsiBearDiv_nl, rsiBullDiv_nl, rsiBearDivHidden_nl

[stochFractalTop, stochFractalBot, stochLow_prev, stochBearDiv, stochBullDiv, stochBearDivHidden, stoc

buySignal = wtCross and wtCrossUp and wtOversold

buySignalDiv = (wtBullDiv or wtBullDiv_add or stochBullDiv or rsiBullDiv)

sellSignal = wtCross and wtCrossDown and wtOverbought
sellSignalDiv = wtBearDiv or wtBearDiv_add or stochBearDiv or rsiBearDiv

length2 = input.int(30, minval=1, group="Shark")
src = input.source(close, title="Source", group="Shark")
mult = input.float(2.0, minval=0.001, maxval=50, title="StdDev", group="Shark")
offset = input.int(0, "Offset", minval = -500, maxval = 500, group="Shark")

HighlightBreaches = input.bool(true, title="Highlight Oversold/Overbought", group="Shark")
bApply25and75 = input.bool(true, title="Apply 25/75 RSI rule", group="Shark")

ema50 = ta.ema(close, 50)
```

```
ema200 = ta.ema(close, 200)
ema400 = ta.ema(close, 400)
ema800 = ta.ema(close, 800)
wapwap = ta.vwap(close)

bTouchedLine = (ema50<high and ema50>low) or (ema200<high and ema200>low) or (ema400<high and ema400>l

upTR = ta.rma(math.max(ta.change(close), 0), 14)
downTR = ta.rma(-math.min(ta.change(close), 0), 14)
rsiM = downTR == 0 ? 100 : upTR == 0 ? 0 : 100 - (100 / (1 + upTR / downTR))

basisWT = ta.sma(rsiM, length2)
devWT = mult * ta.stdev(rsiM, length2)
upperWT = basisWT + devWT
lowerWT = basisWT - devWT

bBelow25 = rsiM < 26
bAbove75 = rsiM > 74

if not bApply25and75
    bBelow25 := true
    bAbove75 := true

b_color = (rsiM > upperWT and bAbove75) ? c_regular_red : (rsiM < lowerWT and bBelow25) ? c_regular_gr

if bTouchedLine and b_color == color.new(color.red, transp=60)
    b_color := c_intense_red

if bTouchedLine and b_color == color.new(color.green, transp=60)
    b_color := c_intense_green

// bgcolor(HighlightBreaches ? b_color : na)


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   FINAL DISPLAY   =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

colorGreen = colorMildGreen
colorRed = colorMildRed

upWick50PercentLarger = close > open and math.abs(high - close) > math.abs(open - close)
downWick50PercentLarger = close < open and math.abs(low - close) > math.abs(open - close)

if (upWick50PercentLarger and rsi > rsiUpperW)
    colorGreen := colorBigGreen
if (downWick50PercentLarger and rsi < rsiLowerW)
    colorRed := colorBigRed

rsiColor = rsi > rsiOB ? colorBigGreen : rsi < rsiOS ? colorBigRed : mfi > 50 ? colorMildGreen : mfi <
rsiPlot = plot(bShowRSI ? rsi : na, "RSI", color=rsiColor, linewidth=3)
midlinePlot = plot(bShowRSI ? 50 : na, "RSI Middle Band", color=color.new(#787B86, 50))

fill(rsiPlot, midlinePlot, 100, 70, top_color = colorBigGreen, bottom_color = colorInvisible, title =
fill(rsiPlot, midlinePlot, 30,  0,  top_color = colorInvisible, bottom_color = colorBigRed, title = "C

plotchar(upThrust and bShowTramp ? 25 : na, title = 'Trampoline', char='T', color = color.white, locat
```

```
plotchar(downThrust and bShowTramp ? 75 : na , title = 'Trampoline', char = 'T', color = color.white,

plotshape((pvsraColor == greenVectorColor) and bShowVector ? 50 : na, title="Vector Candle", location=
plotshape((pvsraColor == redVectorColor) and bShowVector ? 50 : na, title="Vector Candle", location=lc

plotshape(rsi > rsiOB ? 70 : na, title = 'RSI Overbought Square', style = shape.square, color = colorG
plotshape(rsi < rsiOS ? 30 : na , title = 'RSI Oversold Square', style = shape.square, color = colorRe

plotshape(buySignal1 and bShowSqueeze ? 25 : na, title="Squeeze Buy Signal", style=shape.diamond, loca
plotshape(sellSignal1 and bShowSqueeze ? 75 : na, title="Squeeze Sell Signal", style=shape.diamond, lc

plotchar(low <= wlowerBB and close >= wlowerBB and close < open and bShowBB ? 35 : na, char="ß", title
plotchar(high >= wupperBB and close < wupperBB and close > open and bShowBB ? 65 : na, char="ß", title

plotshape(bullCondAlert ? 50 : na,offset=-lookbackRight,title="Regular Bullish Label",text=" Bull",sty
plotshape(bearCondAlert ? 50 : na,offset=-lookbackRight,title="Regular Bearish Label",text=" Bear",sty


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   OUTSIDE LINE PLOTS  =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

//EverColor = EverestDown ? colorBigRed : EverestUp ? colorBigGreen : colorInvisible
vodkaColor = upwards ? colorBigGreen : downwards ? colorBigRed : colorInvisible
imdColor = srcMCD > miMCD ? srcMCD > hiMCD ? colorBigGreen : colorMildGreen : srcMCD < loMCD ? colorBi
KSTColor = KSTTrue ? colorBigGreen : colorBigRed
bullColor1 = bullDown ? colorBigRed : bullUp ? colorBigGreen : colorInvisible
tripColor = tripBuy ? colorBigGreen : tripSell ? colorBigRed : colorInvisible

cciDoubleColor = colorInvisible
if (cci1 > cci2 and cci1 >= 100)
    cciDoubleColor := colorBigGreen
if (cci1 < cci2 and cci1 < -100 and cci1WasGreen)
    cciDoubleColor := colorBigRed

if (cci1 > 0)
    cci1 := cci1 - 200
if (cci1 < 0)
    cci1 := cci1 + 200
cci1 := cci1 + 50
// plot (cci1, color=color.lime)

plot(71, "Vodka Shot", color=vodkaColor, linewidth=2)

newTrip = ((tripBuy and tripSell[0]) or (tripSell and tripBuy[0]))
plot(70, "Triple Supertrend", color=tripColor, linewidth=1)
plot(newTrip ? 70 : na, "Triple Supertrend", color=tripColor, linewidth=5)

plot(69, "Half Trend", color=htColor, linewidth=1)
plot(68, "CCI Double Cross", color=cciDoubleColor, linewidth=1)

plot(31, "Impulse MACD", color=imdColor, linewidth=1)
plot(30, "Bull Rush", color=bullColor1, linewidth=1)
//plot(29, "Everest", color=EverColor, linewidth=2)

plot(bShowWAE ? 50 : na, "Waddah Explosion", color=waeColor, linewidth=thickWaddah)
```

```
bgcolor(buySignalDiv and bShowDivies ? color.new(colorMildGreen, bgTrans-10) : na)
bgcolor(sellSignalDiv and bShowDivies ? color.new(colorMildRed, bgTrans-10) : na)

bgcolor(buySignal and bShowBuySellies ? color.new(colorMildGreen, bgTrans) : na)
bgcolor(sellSignal and bShowBuySellies ? color.new(colorMildRed, bgTrans) : na)


// =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=   ALERTS  =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= /

// RSI divergences
alertcondition(bearCondAlert or bullCondAlert, "RSI Divergence", "RSI Divergence")

bAllActive = (upwards or downwards) and KSTTrue and (bullDown or bullUp) and (tripBuy or tripSell) and
alertcondition(bAllActive, "All Trend Meters Active", "All Trend Meters Active")

alertcondition(buySignalDiv or sellSignalDiv, "Background Divergence", "Background Divergence")
alertcondition(buySignal or sellSignal, "Background Buy/Sell Signal", "Background Buy/Sell Signal")
```
